Name:- Shazim Khot
Roll No:- 25AI57
Div:- SE A
Branch:- AIML

Experiment No. 1A: Introduction to Arrays and Basic Operations.

Aim:-
To study the concept of arrays and implement a menu-driven program for array creation, display, insertion, deletion, and searching.

Arrays in C – Concepts and Theory:-
An **array** is a fundamental data structure in the C language. It represents a group of elements of identical type stored in consecutive memory blocks. Each item is accessed using an index, beginning at 0. Arrays simplify handling multiple values under one variable name, making programs shorter and easier to manage. For instance, instead of declaring 50 separate variables to store student marks, a single array can hold all of them efficiently.

Key Features of Arrays:-
- Elements must share the same data type.
- Memory allocation is contiguous.
- Size is fixed at the time of declaration.
- Direct access to elements via index ensures speed.
- Ideal for managing large sets of related values.

Basic Operations on Arrays:-
1. **Creation** – Defining the array and assigning values to its elements.
2. **Display** – Printing all stored values sequentially.
3. **Insertion** – Adding a new element at a chosen position, shifting existing ones to the right.
4. **Deletion** – Removing an element from a specific index, with subsequent values shifting left.

5. **Searching** – Checking if a particular value exists and reporting its position if found.

Benefits of Using Arrays:-

- Convenient storage of multiple values under one name.
- Quick access through indexing.
- Simplifies program structure and logic.
- Saves memory compared to declaring many variables.
- Essential for implementing algorithms like sorting and searching.

Limitations of Arrays:-

- Size cannot be altered once declared.
- Only one data type can be stored.
- Insertions and deletions require shifting elements, which can be inefficient.

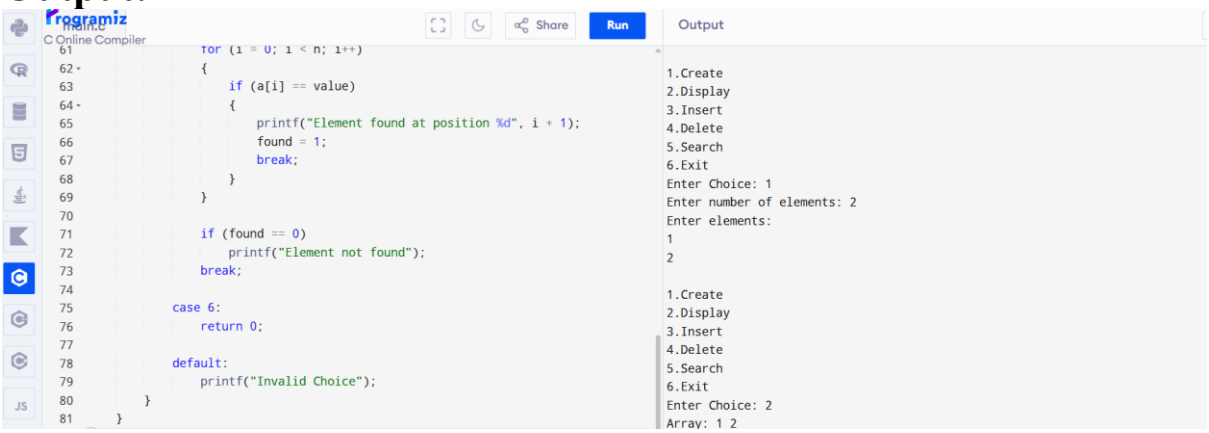
Code:-

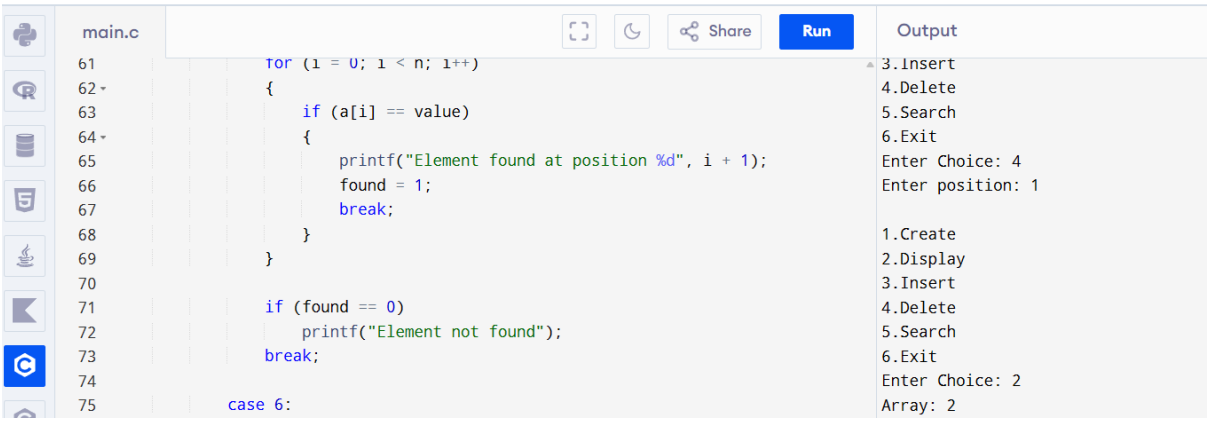
```
#include <stdio.h>

int main()
{
   int a[50], n = 0;
   int choice, i, pos, value, found;

   while (1)
   {

printf("\n1.Create\n2.Display\n3.Insert\n4.Delete\n5.Search\n6.Exit");
      printf("\nEnter Choice: ");
      scanf("%d", &choice);

      switch (choice)
      {
         case 1:
            printf("Enter number of elements: ");
            scanf("%d", &n);
```

```
    printf("Enter elements:\n");
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    break;

case 2:
    printf("Array: ");
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
    break;

case 3:
    printf("\nEnter position: ");
    scanf("%d", &pos);

    printf("Enter value: ");
    scanf("%d", &value);

    for (i = n; i >= pos; i--)
        a[i] = a[i - 1];

    a[pos - 1] = value;
    n++;
    break;

case 4:
    printf("Enter position: ");
    scanf("%d", &pos);

    for (i = pos - 1; i < n - 1; i++)
        a[i] = a[i + 1];

    n--;
    break;

case 5:
```

```
            printf("Enter element to search: ");
            scanf("%d", &value);

            found = 0;

            for (i = 0; i < n; i++)
            {
               if (a[i] == value)
               {
                  printf("Element found at position %d", i + 1);
                  found = 1;
                  break;
               }
            }

            if (found == 0)
               printf("Element not found");
            break;

         case 6:
            return 0;

         default:
            printf("Invalid Choice");
      }
   }
}
```

Output:-





Conclusion:-

The program successfully performs array creation, display, insertion, deletion, and searching using a menu-driven approach in C.

Experiment 1.B

Aim:-

To implement a menu-driven program for performing array operations and understand the working of arrays in C.

Theory and Concepts:-

An array is a data structure that stores multiple values of the same data type using a single variable name. Each element is accessed through its index. Arrays simplify data handling and eliminate the need to declare multiple variables for similar data.

Operations Performed:-

- Create an array
- Display array elements
- Insert an element
- Delete an element
- Search for an element

Advantages:-

- Provides easy access to elements using their index.
- Minimizes the need for multiple variables.
- Efficient for storing and managing similar types of data.

Limitations:-

- Has a fixed size.
- Can store values of only one data type.

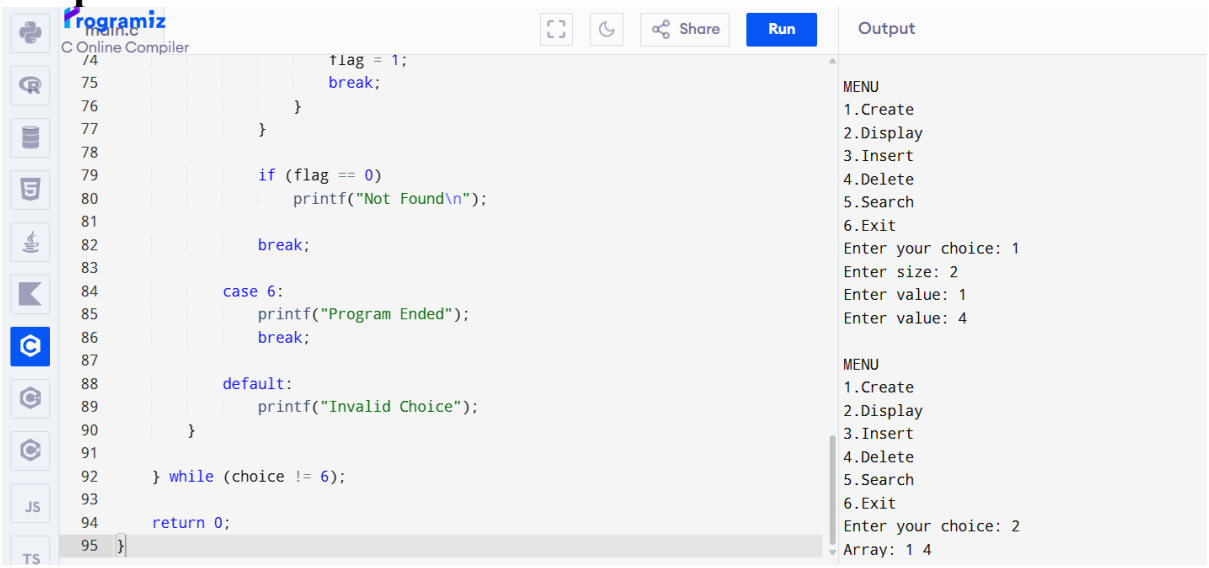
Code:-

```
#include <stdio.h>

int main()
{
   int arr[50], n = 0, choice, i, pos, value, flag;

   do
   {
      printf("\nMENU");
      printf("\n1.Create");
      printf("\n2.Display");
      printf("\n3.Insert");
      printf("\n4.Delete");
      printf("\n5.Search");
      printf("\n6.Exit");

      printf("\nEnter your choice: ");
      scanf("%d", &choice);

      switch (choice)
      {
         case 1:
            printf("Enter size: ");
            scanf("%d", &n);
```

```
    for (i = 0; i < n; i++)
    {
        printf("Enter value: ");
        scanf("%d", &arr[i]);
    }
    break;

case 2:
    printf("Array: ");
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    break;

case 3:
    printf("Enter position: ");
    scanf("%d", &pos);

    printf("Enter value: ");
    scanf("%d", &value);

    for (i = n; i > pos; i--)
        arr[i] = arr[i - 1];

    arr[pos] = value;
    n++;
    break;

case 4:
    printf("Enter position: ");
    scanf("%d", &pos);

    for (i = pos; i < n - 1; i++)
        arr[i] = arr[i + 1];

    n--;
    break;
```

```
            case 5:
                printf("Enter value to search: ");
                scanf("%d", &value);

                flag = 0;

                for (i = 0; i < n; i++)
                {
                    if (arr[i] == value)
                    {
                        printf("Found at index %d\n", i);
                        flag = 1;
                        break;
                    }
                }

                if (flag == 0)
                    printf("Not Found\n");

                break;

            case 6:
                printf("Program Ended");
                break;

            default:
                printf("Invalid Choice");
        }

    } while (choice != 6);

    return 0;
}
```

Output:-



```
                    flag = 1;
                    break;
                }
            }

            if (flag == 0)
                printf("Not Found\n");

            break;

        case 6:
            printf("Program Ended");
            break;

        default:
            printf("Invalid Choice");
    }

} while (choice != 6);

return 0;
}
```

```
MENU
1.Create
2.Display
3.Insert
4.Delete
5.Search
6.Exit
Enter your choice: 1
Enter size: 2
Enter value: 1
Enter value: 4

MENU
1.Create
2.Display
3.Insert
4.Delete
5.Search
6.Exit
Enter your choice: 2
Array: 1 4
```

```
                    flag = 1;
                    break;
                }
            }

            if (flag == 0)
                printf("Not Found\n");

            break;

        case 6:
            printf("Program Ended");
            break;

        default:
            printf("Invalid Choice");
    }

} while (choice != 6);
```

```
Array: 1 4
MENU
1.Create
2.Display
3.Insert
4.Delete
5.Search
6.Exit
Enter your choice: 4
Enter position: 1

MENU
1.Create
2.Display
3.Insert
4.Delete
5.Search
6.Exit
Enter your choice: 2
Array: 1
```

Conclusion:-

This experiment explains the concept of arrays and demonstrates all fundamental array operations through a menu-driven C program. It improves understanding of array manipulation techniques used in programming.